Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	213	LRU near2 pseudo	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L2	255	cache near2 partition	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L3	4211865	way$2 or block$2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L4	4537	least adj2 recent$2 adj2 use$2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L5	213	LRU near2 pseudo	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L6	4537	least adj2 recent$2 adj2 use$2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L7	4597	L5 or L6	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L8	255	cache near2 partition	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L9	4597	L5 or L6	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43

L30	0	l28 and l29	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:51
L31	4090	lru	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:51
L32	5675	l7 or l31	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:51
L33	4	l28 and l32	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:54
L34	2	"5584007".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:55
L35	375642	partition$4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:55
L36	1	l34 and l35	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:55
L37	1	l32 and l36	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:56
L38	1	l25 and l37	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:56
L39	0	l38 and l29	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:56

L10	77	L9 and L8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L11	4211865	way$2 or block$2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L12	77	L9 and L8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L13	72	L12 and L11	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L14	0	dynamica44	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L15	514732	dynamic$4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L16	72	L12 and L11	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L17	514732	dynamic$4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L18	54	L16 and L17	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L19	0	directmap$4 adj2 cache	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:50

L20	603	direct-map$4 adj2 cache	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L21	54	L16 and L17	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L22	603	direct-map$4 adj2 cache	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L23	10	L21 and L22	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:43
L24	5497	graphic$3 adj processor$2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:47
L25	309478	graphic$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:47
L26	2	"5717893".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:47
L27	0	l25 and l26	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:49
L28	4	l8 and l24	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:49
L29	1282	directmap$4 or direct-map$4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/03/03 21:50

US-PAT-NO: 5764945

DOCUMENT-IDENTIFIER: US 5764945 A

TITLE: CD-ROM average access time improvement

DATE-ISSUED: June 9, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Ballard; Clinton L.	Bremerton	WA	98312	N/A

APPL-NO: 08/ 664454

DATE FILED: June 17, 1996

PARENT-CASE:

This is a Continuation of application Ser. No. 08/194,104 filed Feb. 9, 1994; and Ser. No. 08/585,808, filed Jan. 16, 1996.

US-CL-CURRENT: 711/113, 711/118 , 711/138

ABSTRACT:

A cache for improving access to optical media includes a primary cache comprising RAM and a secondary cache comprising a portion of hard disk memory. Multiple aspects of the invention are defined: (1) Cache data discrimination: Discrimination methodology is implemented for determining when data should not be cached. Under certain conditions, caching of data is less likely to improve access time. (e.g., when the transfer rate is already exceeding a critical sustained throughput rate; when an estimated time to complete a CD-ROM data request is within a specific percentage of the estimated time to complete a hard drive disk request). (2) Secondary cache fragmentation avoidance: To keep the access time to secondary cache faster than the access time to the optical media, fragmentation of the secondary cache (i.e., hard disk) is minimized. To do so, constraints are imposed: (i) an entire CD-ROM request is stored in contiguous sectors on the hard drive; (ii) sequential CD-ROM requests to adjacent sectors of CD-ROM are concatenated on the hard drive; (iii) data redundancy is permitted). (3) Alternative update methodologies: Cache updates are performed in sequence or in parallel to primary and secondary cache depending upon the embodiment. (4) Data integrity: Integrity of data stored in non-volatile secondary cache is maintained for a substantial portion of secondary cache through power failures, shutdowns and media swaps.

4 Claims, 13 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 9

---------- KWIC ---------

Detailed Description Text - DETX (24):

If cache 100 is not used for this data transfer, then at step 118 data is transferred from CD-ROM to RAM 24 or a processing unit for processing (e.g., to

CPU 20, **graphics processor** 32, video processor 34, or audio processor 38). The request is then complete at step 120.

Detailed Description Text - DETX (26):

In one embodiment, partial hits are supported. In such case, the remainder of the data request, if any, is processed to access the remaining data from the CD-ROM 70 at step 134. At step 136, such data is then combined with the data from primary cache 102 and secondary cache 104, if any. If partial hits are not supported, the entire data request is passed to check secondary cache 104 or CD-ROM 70 to complete the request. The cache 100 is then updated to store the data request at step 138. According to a preferred embodiment, primary and secondary caches 102, 104 are updated in parallel. Conventional hashing tables, binary trees or balanced trees are alternative methods of maintaining cross-referencing between CD-ROM and cache locations. According to a preferred embodiment, when cache 102 or 104 fills up, data is discarded from the respective cache using a first-in first-out criteria. Alternatively, a **least-recently-used** or usage-count criteria is implemented instead. The request is then complete at step 120.

Detailed Description Text - DETX (40):

FIG. 12 shows a two-stage update process embodiment. At step 230, the primary cache look-up tables are tested to determine whether there is enough room in primary cache 102 to store the data request. If enough room (i.e., yes), then at step 232 the look-up table (e.g., hashing table) structures are updated for enabling future access to the data. If not enough room (i.e., no), then at step 234 some data in primary cache 102 is discarded to make room for the request. According to alternative embodiments, first-in first-out, **least-recently-used** or usage-count criteria are used to determine which data to discard. Then, at step 236 secondary cache look-up tables (e.g., hashing tables) are tested to determine whether there is enough room in secondary cache 104 to store the request. If not enough room (i.e., no), then at step 238 some data is discarded from secondary cache 104 to make room. Again, according to alternative embodiments first-in first-out, **least-recently-used** or usage-count criteria are used to determine which data to discard. After data is discarded (step 238) or directly after testing the space availability in secondary cache (step 236 answered yes), the request is tested at step 240 to determine whether the request exceeds the updatable area (i.e., crosses over into write-protected area). If exceeded (i.e., yes), then at step 242 the request is queued to update the non-volatile directory (e.g., portion of look-up table structures) with new divisions (i.e., boundaries between updatable and write-protected areas). After queuing (step 242) or directly after testing (i.e., step 240 answered no), the secondary cache data structures are updated at step 244 to enable future access to the data. Then, at step 232 the primary cache look-up data structures are updated to enable future access to the data. Cache update processing for the data request is then complete at step 246.

Detailed Description Text - DETX (69):

For the coded embodiment, secondary cache 104 is formed as a DOS file. A problem of using the conventional DOS file system is that fragmentation of the cache file can occur. This would ultimately lead to the need for defragmenting the file system, which could take many minutes. If defragmentation of the cache file were performed, each multiple sector cd rom request would end up becoming a large number of different disk requests. Since each disk request involves a seek of the head and a rotational latency, which typically combines to take 20 milliseconds, the overall throughput from the disk drive will gradually degrade to become worse than the CD-ROM drive. In one embodiment an entire DOS **partition is allocated to the cache** file. Such approach is

impractical, however, for upgrading existing systems which already have all of the physical disk space allocated to existing partitions. In the implemented embodiment, initial fragmentation (created by the operating system) is accepted, but additional fragmentation is avoided. Avoiding additional fragmentation is achieved by allocating disk space via the file system, then write protecting the space so it will not be moved around by the operating system. By setting the DOS attributes for the cache file to be SYSTEM and HIDDEN and READ-ONLY, DOS does not further fragment the cache file. It even prevents defragmenters from moving the allocated physical sectors around.

US-PAT-NO: **5584007**

DOCUMENT-IDENTIFIER: US 5584007 A

TITLE: Apparatus and method for discriminating among data to be stored in cache

DATE-ISSUED: December 10, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Ballard; Clinton L.	Bremerton	WA	N/A	N/A

APPL-NO: 08/ 194104

DATE FILED: February 9, 1994

US-CL-CURRENT: 711/113, 711/122 , 711/138

ABSTRACT:

A cache for improving access to optical media includes a primary cache comprising RAM and a secondary cache comprising a portion of hard disk memory. Multiple aspects of the invention are defined: (1) Cache data discrimination: Discrimination methodology is implemented for determining when data should not be cached. Under certain conditions, caching of data is less likely to improve access time. (e.g., when the transfer rate is already exceeding a critical sustained throughput rate; when an estimated time to complete a CD-ROM data request is within a specific percentage of the estimated time to complete a hard drive disk request). (2) Secondary cache fragmentation avoidance: To keep the access time to secondary cache faster than the access time to the optical media, fragmentation of the secondary cache (i.e., hard disk) is minimized. To do so, constraints are imposed: (i) an entire CD-ROM request is stored in contiguous sectors on the hard drive; (ii) sequential CD-ROM requests to adjacent sectors of CD-ROM are concatenated on the hard drive; (iii) data redundancy is permitted). (3) Alternative update methodologies: Cache updates are performed in sequence or in parallel to primary and secondary cache depending upon the embodiment. (4) Data integrity: Integrity of data stored in non-volatile secondary cache is maintained for a substantial portion of secondary cache through power failures, shutdowns and media swaps.

14 Claims, 13 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 9

---------- KWIC ---------

US Patent No. - PN (1):
5584007

Brief Summary Text - BSTX (10):
Programmers typically optimize information layout on a CD-ROM to achieve the current CD-ROM access times. Straight sequential access of optical media as

during video playback or audio playback typically can occur within desired specifications to provide a user with acceptable visual and sound quality. When random seeks, however, are interspersed with full-motion video and/or sound, performance (i.e., average access time) degrades. For typical multimedia and hypermedia applications, audio/video data is interspersed with program and/or **graphics** data. In a conventional CD-ROM encyclopedia application, a user may spend 90% of the time accessing program and **graphics** data and just 10% of the time accessing audio-video or animation data. (The CD-ROM itself may comprise 50% data and 50% video/audio due to the storage intensiveness of video/audio.) A conventional CD-ROM drive meeting multimedia specifications requires a sustained throughput rate of 150 KB/second. It is the occasions when random seeks are interspersed with the full motion video or animation that problems typically arise. Using a conventional RAM cache of approximately 256 KB, the cache will fill with audio-video data very quickly. There already is significant contention for the RAM resources by the operating system. Using all of the available cache for maintaining full motion video would be unacceptable. Accordingly, a larger high-capacity cache is needed.

Brief Summary Text - BSTX (23):

One advantage of this invention is that average data access times for optical media can be improved for many microcomputer multimedia, hypermedia, animation and other video, audio-video and **graphical** applications. Another advantage is that the cache structure can be implemented with existing resources on a microcomputer by allocating a portion of system RAM and a portion of a user's hard drive. Cache implementation software defines the cache structure and controls operation. By implementing a high-capacity cache filled up over a period of hours or weeks and maintaining the integrity of cache data across shutdowns and CD-ROM swaps, performance improvements are maintained over the long term, (rather than following a latent period of use after each "power on" or "media swap"). Accordingly, a cost-efficient, technically-effective cache is implemented for improving access times for optical storage media.

Detailed Description Text - DETX (4):

A local bus 14 is linked to the processor bus 12 via a local bus interface 30. Exemplary local busses are the video local (VL) or VESA-standard bus, the peripheral component interface (PCI) bus and the NU-BUS. The PCI bus, for example, may couple up to 10 peripheral devices. Illustrated are a **graphics** controller 32, video processor 34, video capture/output card 36 and a sound card 38. Such peripherals are used in multimedia and audio-video production systems. Speakers 40 and a microphone 42 are linked to the sound card. A camera 44 (e.g., camcorder), VCR 46 and TV 48 are linked to the video capture/output card 36. The video subsystems 32, 34, 36 typically share a local memory resource (i.e., frame buffer or video RAM) 50. Information is passed to a display 52 from the video subsystems and shared memory 50 via a video DAC 54.

Detailed Description Text - DETX (11):

According to a preferred embodiment, the primary cache 102 provides 0.5 MB to 2 MB of storage, while the secondary cache 104 provides at least 10MB of storage (e.g., 20 MB to 140 MB). For the structure of FIG. 3, the secondary cache 104 is formed from an area of hard drive 60. For a DOS-based machine, such area is formed as a DOS file or a DOS **partition**. According to other operating systems, the area may be formed by a file, object or other operating system or user mechanism for dedicating address space to the secondary cache 104. Regardless of the operating system, it is preferable when possible that physical address space be allocated as secondary cache 104 so that the cache

implementation software can avoid fragmentation of stored data.

Detailed Description Text - DETX (22):

The optical media cache 100 is implemented for a CD-ROM 70 according to a preferred embodiment. FIG. 3 shows a partial block diagram of a microcomputer system implementing the CD-ROM cache 100. Application programs requiring data from CD-ROM 70 are executed by CPU 20. The CPU 20 controls data flow by directing video data to a video processor 34, audio data to a sound card 38 and **graphics** data to a **graphics** controller 32. Typically, the CPU 20 processes conventional program data. Accordingly, there may be four data streams (i.e., program data, video data, audio data and **graphics** data). Full motion video, for example, may require a throughput of 30-60 MB per second for an application with 24-bit color, 2.3 MB/frame (without compression) and updating at 30 frames per second (fps). Typically, the video data is compressed to achieve the sustained throughput necessary for clear full-motion video. Sound may require another 10 MB per second throughput before compression. To achieve the necessary sustained throughput, companies are optimizing performance in many areas. FIG. 5 shows the layout of a conventional CD-ROM 106. A storage track 108 spirals around and in toward the center of the disk. To optimize performance, programmers estimate frequently accessed portions and store it near the center of the disk 106. Data near the center can be accessed more quickly than data at the outer edge. In addition video compression and audio compression techniques are frequently used so that less data is needed to develop a video image or audio sound clip. The contribution of this invention is to provide an optical media cache for storing frequently accessed data in a faster storage medium.

Detailed Description Text - DETX (24):

If cache 100 is not used for this data transfer, then at step 118 data is transferred from CD-ROM to RAM 24 or a processing unit for processing (e.g., to CPU 20, **graphics** processor 32, video processor 34, or audio processor 38). The request is then complete at step 120.

Detailed Description Text - DETX (26):

In one embodiment, partial hits are supported. In such case, the remainder of the data request, if any, is processed to access the remaining data from the CD-ROM 70 at step 134. At step 136, such data is then combined with the data from primary cache 102 and secondary cache 104, if any. If partial hits are not supported, the entire data request is passed to check secondary cache 104 or CD-ROM 70 to complete the request. The cache 100 is then updated to store the data request at step 138. According to a preferred embodiment, primary and secondary caches 102, 104 are updated in parallel. Conventional hashing tables, binary trees or balanced trees are alternative methods of maintaining cross-referencing between CD-ROM and cache locations. According to a preferred embodiment, when cache 102 or 104 fills up, data is discarded from the respective cache using a first-in first-out criteria. Alternatively, a **least-recently-used** or usage-count criteria is implemented instead. The request is then complete at step 120.

Detailed Description Text - DETX (40):

FIG. 12 shows a two-stage update process embodiment. At step 230, the primary cache look-up tables are tested to determine whether there is enough room in primary cache 102 to store the data request. If enough room (i.e., yes), then at step 232 the look-up table (e.g., hashing table) structures are updated for enabling future access to the data. If not enough room (i.e., no), then at step 234 some data in primary cache 102 is discarded to make room for

the request. According to alternative embodiments, first-in first-out, **least-recently-used** or usage-count criteria are used to determine which data to discard. Then, at step 236 secondary cache look-up tables (e.g., hashing tables) are tested to determine whether there is enough room in secondary cache 104 to store the request. If not enough room (i.e., no), then at step 238 some data is discarded from secondary cache 104 to make room. Again, according to alternative embodiments first-in first-out, **least-recently-used** or usage-count criteria are used to determine which data to discard. After data is discarded (step 238) or directly after testing the space availability in secondary cache (step 236 answered yes), the request is tested at step 240 to determine whether the request exceeds the updatable area (i.e., crosses over into write-protected area). If exceeded (i.e., yes), then at step 242 the request is queued to update the non-volatile directory (e.g., portion of look-up table structures) with new divisions (i.e., boundaries between updatable and write-protected areas). After queuing (step 242) or directly after testing (i.e., step 240 answered no), the secondary cache data structures are updated at step 244 to enable future access to the data. Then, at step 232 the primary cache look-up data structures are updated to enable future access to the data. Cache update processing for the data request is then complete at step 246.

Detailed Description Text - DETX (69):

For the coded embodiment, secondary cache 104 is formed as a DOS file. A problem of using the conventional DOS file system is that fragmentation of the cache file can occur. This would ultimately lead to the need for defragmenting the file system, which could take many minutes. If defragmentation of the cache file were performed, each multiple sector cd rom request would end up becoming a large number of different disk requests. Since each disk request involves a seek of the head and a rotational latency, which typically combines to take 20 milliseconds, the overall throughput from the disk drive will gradually degrade to become worse than the CD-ROM drive. In one embodiment an entire DOS **partition** is allocated to the cache file. Such approach is impractical, however, for upgrading existing systems which already have all of the physical disk space allocated to existing **partitions**. In the implemented embodiment, initial fragmentation (created by the operating system) is accepted, but additional fragmentation is avoided. Avoiding additional fragmentation is achieved by allocating disk space via the file system, then write protecting the space so it will not be moved around by the operating system. By setting the DOS attributes for the cache file to be SYSTEM and HIDDEN and READ-ONLY, DOS does not further fragment the cache file. It even prevents defragmenters from moving the allocated physical sectors around.